Hindawi
Journal of Electrical and Computer Engineering
Volume 2023, Article ID 1071142, 17 pages
https://doi.org/10.1155/2023/1071142

Research Article

Performance Evaluation of Conventional and Neural Network-Based Decoder for an Audio of Low-Girth LDPC Code

Dharmeshkumar Patel[1] **and Ninad Bhatt**[2]

[1]*Electronics & Communication Engineering, Dr. S. & S. S. Ghandhy Government Engineering College, Gujarat Technological University, Ahmedabad, Gujarat, India*
[2]*Electronics & Communication Engineering, Gujarat Technological University, Ahmedabad, Gujarat, India*

Correspondence should be addressed to Dharmeshkumar Patel; lec_dharmesh@gtu.edu.in

Received 7 January 2023; Revised 25 July 2023; Accepted 7 September 2023; Published 27 October 2023

Academic Editor: Yang Li

Noise in a communication system degrades the signal level at the receiver, and as a result, the signal is not properly recovered or eliminated at the receiver side. To avoid this, it is necessary to modify the signal before transmission, which is achieved using channel coding. Channel coding provides an opportunity to recover the noisy signal at the receiver side. The low-density parity-check (LDPC) code is an example of a forward error correcting code. It offers near Shannon capacity approaching performance; however, there is a constraint regarding high-girth code design. When the low-girth LDPC code is decoded using conventional methods, an error floor can occur during iterative decoding. To address this issue, a neural network (NN)-based decoder is utilized to overcome the decoding problem associated with low-girth codes. In this work, a neural network-based decoder is developed to decode audio samples of both low- and high-girth LDPC codes. The neural network-based decoder demonstrates superior performance for low-girth codes in terms of bit error rate (BER), peak signal-to-noise-ratio (PSNR), and mean squared error (MSE) with just a single iteration. Audio samples sourced from the NOIZEUS corpus are employed to evaluate the designed neural network. Notably, when compared to a similar decoder, the decoder developed in this study exhibits an improved bit error rate for the same signal-to-noise ratio.

1. Introduction

The communication system consists of a transmitter and a receiver, with the channel coder playing a crucial role in this system. The channel coder is used as an encoder at the transmitter side and as a decoder at the receiver side. It provides reliability to both wired and wireless channels in the communication system and is commonly referred to as error correcting codes. Channel coding can be broadly classified into two categories: forward error correcting code and backward error correcting code. One example of a forward error correcting code is the low-density parity-check (LDPC) code, which is also known as a capacity approaching code [1]. LDPC codes greatly enhance the reliability of communication [2]. Forward error correcting codes add additional bits to the information bits in order to facilitate the recovery of the original information at the receiver side [3]. Channel coders find significant applications in deep-space communication, satellite communication, and wireless communication for data transmission [4]. LDPC codes can be constructed in two ways: random and structural. Short length quasi cyclic-LDPC codes, which are widely used in audio broadcasting systems, are an example of structured LDPC codes [5]. LDPC codes were introduced by R. Gallager in his Ph.D. thesis and have proven to be suitable channel coding schemes for high throughput transmission. They are also considered as channel coding schemes for the IEEE 802.11ax system [6]. The encoder of an LDPC code can be based on either linear time encoding or Gauss–Jordan elimination methods, while the decoder can be either hard decision- or soft decision-based.

The computational complexity of a soft decision decoder is generally higher than that of a hard decision decoder. Examples of soft decision decoders include belief propagation and min-sum decoders. The belief propagation decoder incurs higher computational complexity due to operations such as hyperbolic tangent calculations, multipliers, adders, and comparators. In contrast, the min-sum decoder eliminates the need for multipliers and hyperbolic tangent functions in the decoding process, reducing its computational complexity [7]. For an (N, K) regular LDPC code with a column weight of w_c and a row weight of w_r, the decoding process requires approximately $2w_rN$ addition operations and $2w_cN$ comparison operations [5]. Conventional decoding performs optimally when the constructed code is cycle-free, highlighting the importance of reducing the impact of cycles [2].

Understanding the decoding of an LDPC code can be facilitated by analysing its Tanner graph. The Tanner graph, introduced by R. M. Tanner, illustrates the connections between variable nodes and check nodes, revealing how and where messages are passed between them. The length of the smallest cycle in the graph is referred to as the girth of the code [8]. A cycle in a Tanner graph represents a path that alternates between check nodes and variable nodes, starting and ending at the same node, without any node (except the initial and final nodes) appearing more than once [8]. The presence of short cycles, particularly those with a girth of 4, in the parity-check matrix of LDPC codes can hinder convergence. Such codes may not converge well to valid codewords, as the short girth reduces the independence of transmitted messages during the decoding process, leading to a higher error floor [9, 10]. As cycle length decreases, the frequency of incorrect information being recycled increases, making error correction more challenging [2]. The presence of short cycles in low-density parity-check codes can result in performance degradation [2]. A variable node set is referred to as a stopping set if all its neighbours are connected to this set at least twice. Eliminating all stopping sets and trapping sets to construct LDPC codes can be challenging [10]. To improve decoding performance and reduce the error floor, algorithms such as the progressive edge growth (PEG) algorithm and hill climbing algorithm are used to increase the degree of variable nodes [2].

Neural networks have gained significant attention and have been successfully applied in various domains, including channel coding in communication systems, replacing conventional systems in recent years [11]. Neural network-based decoders have shown competitive performance in terms of bit error rate (BER) when compared to conventional decoding methods. Particularly for high-density parity-check codes, neural network-based decoders achieve BER close to that of maximum likelihood decoders and comparable to standard belief propagation decoders [12]. In addition, neural network-based decoders demonstrate faster convergence compared to traditional decoding algorithms, requiring fewer iterations to achieve accurate decoding results [13]. There are various neural network architectures available for solving engineering problems, including those related to channel coding [14]. Different network structures such as feedforward neural networks (FFNNs), recurrent neural networks (RNNs), or convolutional neural networks (CNNs) can be employed based on the specific requirements and characteristics of the coding scheme. Neural networks offer an alternative approach to optimize conventional methods or address problems where traditional mathematical formulas may not be applicable. They leverage the power of machine learning and pattern recognition to tackle complex problems in channel coding. The approach of deep learning-based neural network design provides comparable performance in both high-density and low-density parity-check codes when compared to the belief propagation (BP) decoding algorithm. This approach requires fewer iterations compared to conventional decoding methods [15]. The most common network used in channel coding is a multilayer perceptron (MLP) or feedforward neural network (FFNN) or deep neural network (DNN). In the neural network-based decoding approach, the probabilistic matrix calculations between bit nodes and check nodes, which are required in conventional decoding methods, are eliminated [16]. The neural network-based decoder can either be a standalone FFNN, RNN, and CNN or a combination of a conventional decoder and neural network.

Various works have been carried out on neural network-based decoder design such as the architecture described in the study in [15] introduces a novel approach using the offset min-sum algorithm in combination with a neural network. In this architecture, the offset value in the min-sum algorithm is treated as a learnable parameter of the neural network. The decoder based on this architecture demonstrates a 1 dB improvement in performance compared to the traditional belief propagation decoder. The authors in [15] further compare the results with a neural network (NN) decoder that utilizes multiplicative weights. The NN-based offset min-sum decoder achieves a coding gain that is 0.1 dB lower than the NN-based belief propagation decoder. The performance comparison of the conventional belief propagation, NN-based offset min-sum, and NN-based belief propagation decoders is presented in Figure 1. The obtained bit error rate (BER) results highlight the superior performance of the proposed architecture in the study in [15]. In the decoder proposed by authors in [15], the ReLU activation function is used during the check node process calculation. This choice of activation function is likely based on its ability to introduce nonlinearities and enhance the decoder's learning and decision-making capabilities. Overall, the architecture presented in study in [15] combines the offset min-sum algorithm with a neural network, leveraging learnable parameters and the ReLU activation function to achieve improved decoding performance compared to traditional approaches.

The NN decoder proposed by authors in [17] is based on a neural belief decoder and utilizes learning weights at the edges of variable and check nodes. The architecture of the decoder consists of 10 hidden layers. During training, the network is provided with all-zero codewords using a batch size of 20 for each signal-to-noise ratio (SNR). The SNR values are specified but not provided in the given information. The NN decoder is evaluated on BCH codes with

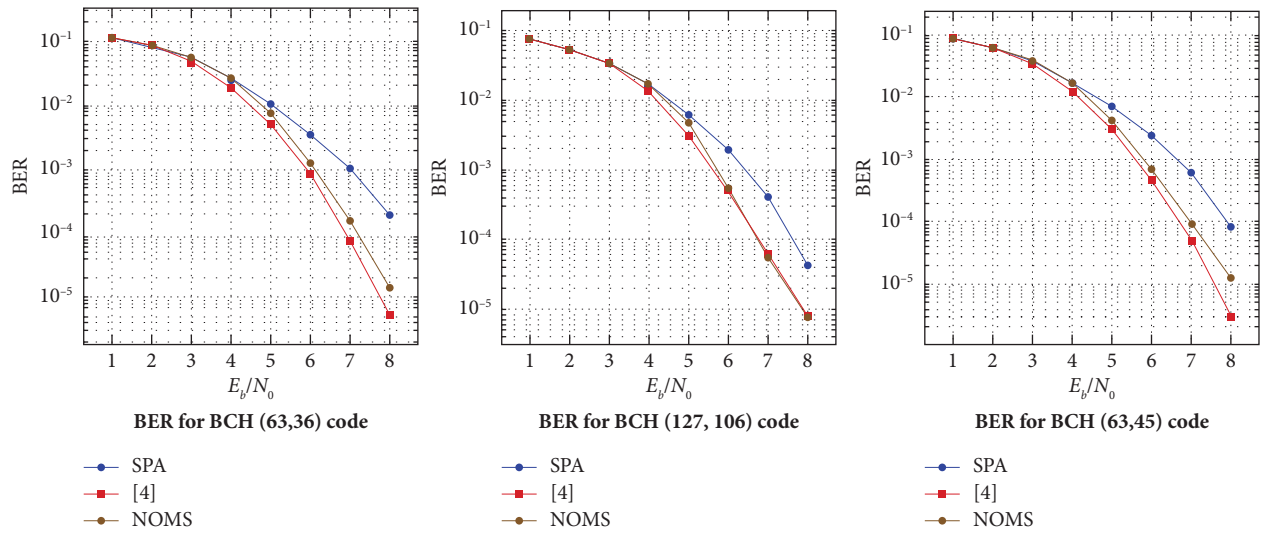

Figure 1: Comparative analysis of BER of conventional and NN decoder for BCH code [15].

different lengths and rates, including (63, 36), (63, 45), and (127, 106). The RMSPROP learning algorithm with a learning rate of 0.001 is employed for training. The results obtained from the experiments show that the performance of the NN decoder for short codes is similar to that of the ML (maximum likelihood) decoder. For large codes, the NN decoder achieves a coding gain of 0.75 dB with fewer iterations compared to the traditional belief propagation (BP) decoder. The specific iteration numbers and SNR values are not mentioned in the given information. The obtained results are presented in Figure 2, showcasing the performance of the NN decoder for the mentioned BCH codes. It demonstrates the effectiveness of the proposed architecture in achieving improved decoding performance compared to the traditional BP decoder. In summary, the NN decoder introduced in study in [17] based on a neural belief decoder shows promising results for decoding BCH codes. It achieves competitive performance compared to the ML decoder for short codes and provides a significant coding gain and reduced iteration count for large codes when compared to the traditional BP decoder.

The NN decoder proposed in the study in [18] is designed to decode both random and structured polar codes. The decoder demonstrates performance similar to the MAP (maximum a posteriori) decoder. Notably, the proposed decoder exhibits the ability to generalize well to structured codes compared to random codes. The design of the NN decoder begins with three hidden layers, with 128, 64, and 32 neurons in each layer, respectively. A sigmoid activation function is used for the neurons [18]. The training process involves utilizing 16 bit code lengths with 20,000 samples for SNR ranging from 0 to 5 dB.

The obtained results for the bit error rate (BER) of both structured and random codes are shown in Figure 3 [18]. The results indicate that the performance of the structure codes is comparable to that of the MAP decoder. Moreover, the NN decoder achieves this performance with a smaller number of epochs compared to the randomly constructed codes. It is worth noting that during training, when higher SNR values are provided, the model tends to learn the structure of the code. Conversely, when training is conducted at lower SNR values, the model focuses on learning about the noise characteristics [18]. In summary, the NN decoder proposed in study in [18] demonstrates effective decoding of both random and structured polar codes. The decoder achieves performance similar to the MAP decoder and exhibits better generalization for structured codes. The model's ability to learn the code structure and adapt to different noise levels is observed, leading to improved decoding accuracy.

In study in [19], three types of decoders are proposed: neural belief, neural min-sum, and neural recurrent neural network (RNN). These decoders are evaluated using BCH codes of different lengths: (63, 36), (63, 45), (127, 64), and (127, 99). The decoders are trained using the RMSPROP learning algorithm with learning rates of 0.001 and 0.0003. In the neural belief propagation decoding, weights are assigned at the edges of the Tanner graph, while in the neural min-sum decoding, the weights are added or multiplied in the check node process, depending on whether it is an offset min-sum or normalized min-sum network, respectively [19]. The decoding performance of both conventional and NN decoders is obtained and compared, as shown in Figure 4 [19]. It is observed that the performance of the RNN decoder is 0.2 dB better than that of the feedforward neural network (FFNN) decoder. Furthermore, the performance of the reduced cycle parity-check matrix is 0.6 to 1.0 dB better than the NN decoder performance [19]. In summary, the decoders proposed in the study in [19], including the neural belief, neural min-sum, and neural RNN, are evaluated on BCH codes. The RNN decoder shows improved performance compared to the FFNN decoder, achieving a gain of 0.2 dB. In addition, the performance of the reduced cycle parity-check matrix is significantly better, with a gain of 0.6 to 1.0 dB compared to the NN decoder performance.

In the study in [20], a deep neural network decoder for a polar code with a length of 16 bits and a half rate is

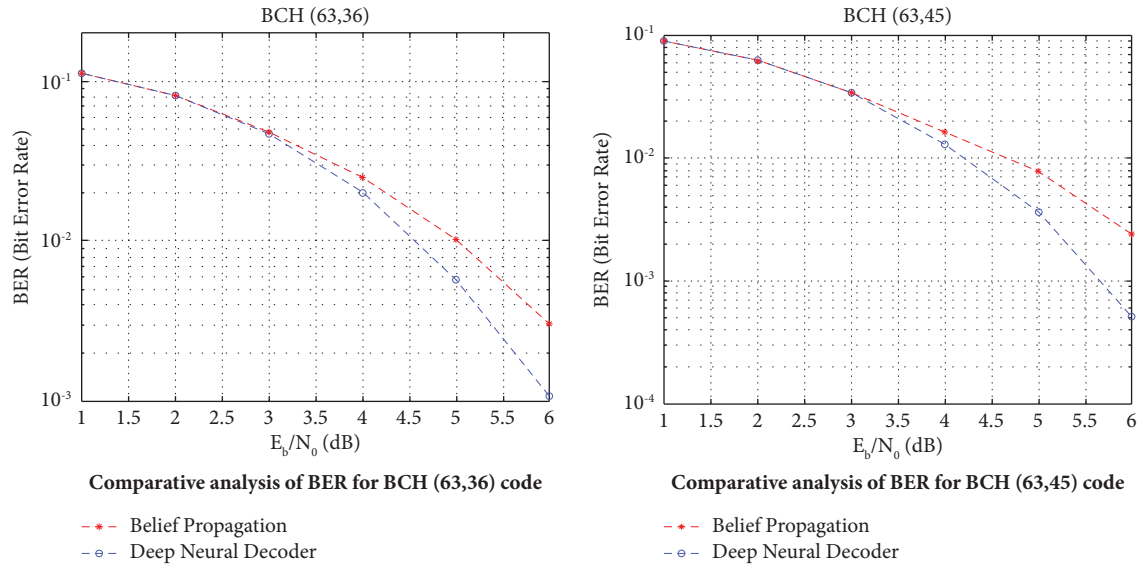

Figure 2: Comparative analysis of BER of conventional and NN decoder for BCH code [18].

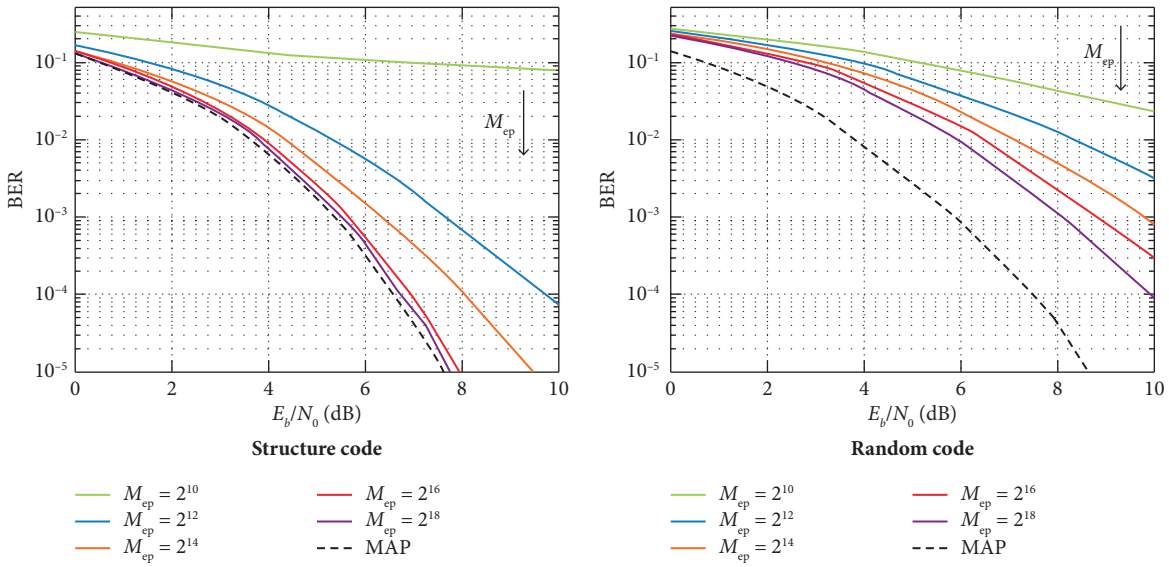

Figure 3: BER performance of structure and random constructed code [18].

proposed by the author. The decoder is tested with different parameters of the neural network, such as the number of hidden layers, the number of neurons, and the type of activation function. The network architecture is implemented using the Keras as the frontend and the Theano as the backend. It consists of three hidden layers with 128-64-32 neurons, an input layer with 16 neurons, and an output layer with 8 neurons. During the training process, the network is trained using all possible code words of length 16 bits; hence, the network is trained by 2^{16} codeword. First, the network is tested with different activation functions using 128-64-32 neurons in the hidden layers. Then, it is tested with two layers using different numbers of neurons in each layer. The results in Figure 5 demonstrate that the rectified linear unit (ReLU) activation function outperforms other activation functions. In addition, increasing the number of hidden layers and the number of neurons in the network improves the performance of the decoder. In summary, the authors in [20] introduce a deep neural network decoder for polar codes of 16 bits and a half rate. The decoder is tested with different neural network parameters, and the results show that using ReLU activation function and increasing the number of hidden layers and neurons improve the decoder's performance.

2. Methodology of Construction, Encoding, and Decoding

2.1. Construction of a Parity-Check Matrix. The first step in LDPC-based coding is to construct a parity-check matrix. The parity-check matrix, as shown in (1), represents the structural type of an LDPC code. It is used for decoding the

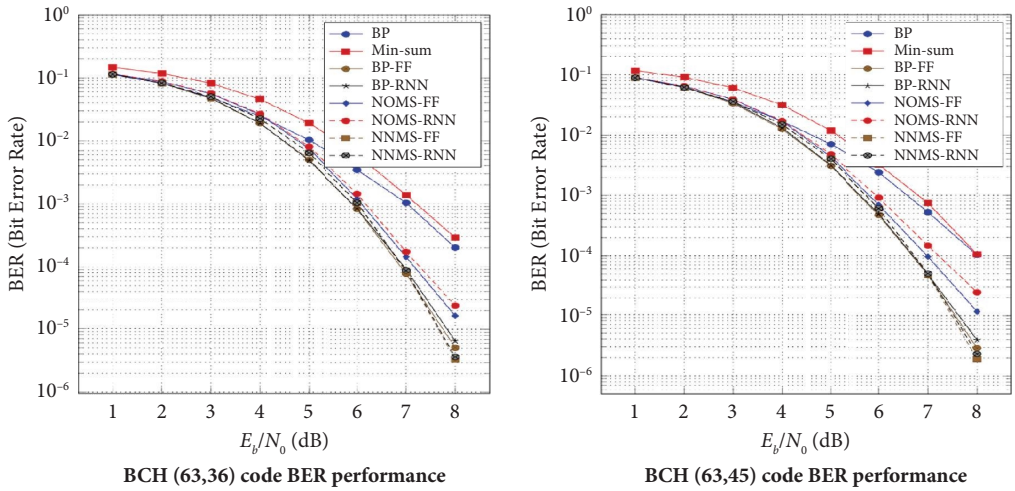

Figure 4: BER performance of BCH (63, 36) and BCH (63, 45) codes [19].

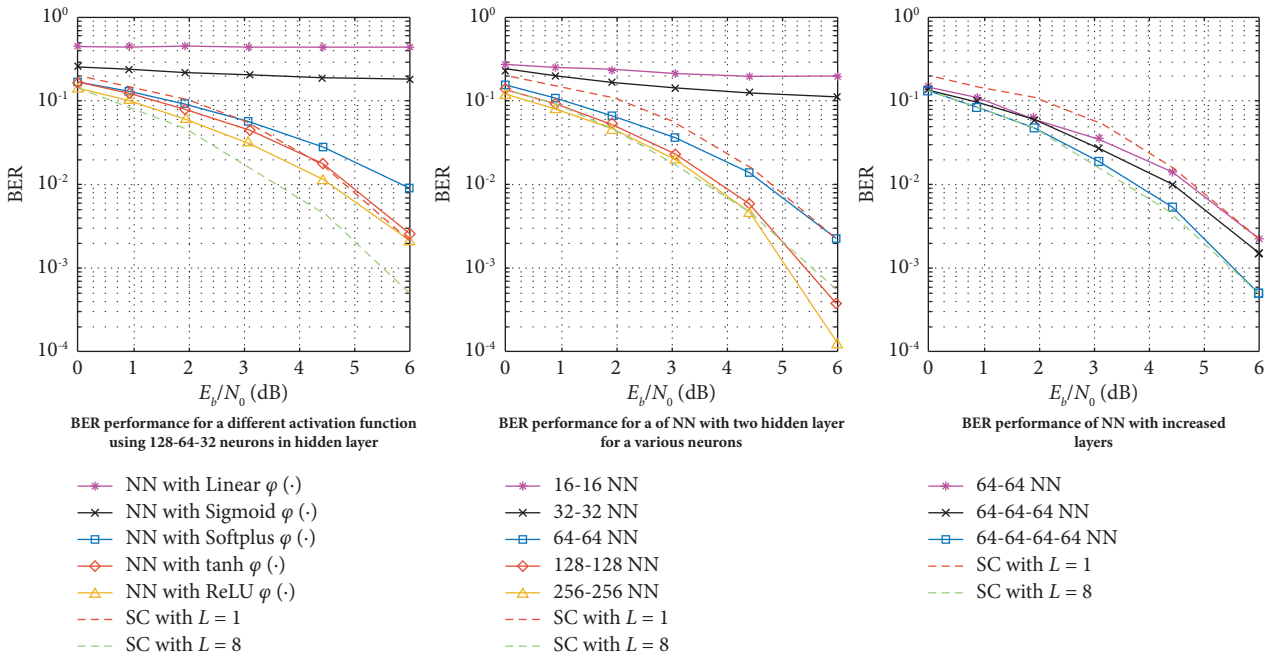

Figure 5: BER performance of NNs for a different activation function, number of neurons, and different hidden layers [20].

code and also serves as the basis for creating a generator matrix for encoding the code.

$$H1 = \begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 & I_6 \\ I_7 & I_8 & I_9 & I_{10} & I_{11} & I_{12} \\ I_{13} & I_{14} & I_{15} & I_{16} & I_{17} & I_{18} \end{bmatrix}. \quad (1)$$

In the LDPC code's parity-check matrix H1, each submatrix is a circularly shifted identical matrix with a size of 9×9. Therefore, the overall size of the parity-check matrix is 27×54, where 54 represents the code length. Equation (1) represents a specific parity-check matrix, which is designed to have a low girth. The matrix H1 is also used in previous work [21] for decoding of an image. It is a type of low-girth matrix as it contains many short cycles.

2.2. Encoding of an Audio. The second step in LDPC-based coding involves encoding the given message. The encoder is employed at the transmitter side of the communication system. The parity-check matrix H1, as shown in (1), is utilized to construct the generator matrix G. The construction of G involves extracting the parity bits using the Gauss–Jordan elimination method and appending an identity matrix to form G. The generator matrix can be represented as $G = [I|P^T]$ or $G = [P^T|I]$, where P^T represents the transposed matrix of parity bits and I represents the identity matrix.

The audio signal, in the form of wave files, is converted into 9644 frames, with each frame consisting of 16 bits. This results in a total of 154,304 bits for the audio message. These message bits are divided into frames of 27 bits each. The

generator matrix G is then used to encode these frames, resulting in 5715 frames of 54 bit codes. After encoding, the encoded audio message is transmitted through the additive white Gaussian noise (AWGN) channel. Before transmission, the BPSK modulation technique is applied to the encoded bits to convert them into symbols that can be transmitted over the channel. BPSK modulation maps each bit to a specific phase of the carrier signal, typically represented as +1 and −1, corresponding to 0 and 1 bits, respectively.

Therefore, the encoded audio message, in the form of BPSK-modulated symbols, is transmitted through the AWGN channel for further processing and reception at the receiver side.

2.3. Conventional Method of Decoding. The third step in LDPC-based coding is the decoding process, which takes place at the receiver side of the communication system. In the current work, the decoding process utilizes the min-sum iterative decoding algorithm as a conventional decoder. During the iterative decoding process, calculated messages are exchanged iteratively between variable nodes and check nodes. The decoding algorithm aims to iteratively estimate the original transmitted message by minimizing the difference between the received signal and the decoded signal. This iterative approach allows for error correction and improves the accuracy of the decoded message. The messages passed between variable nodes and check nodes contain information about the reliability and likelihood of the transmitted bits. These messages are updated and refined through multiple iterations, gradually improving the accuracy of the decoded message. Overall, the decoding process in LDPC-based coding involves the iterative exchange of messages between variable nodes and check nodes to iteratively estimate the original transmitted message, ultimately improving the reliability and accuracy of the decoding process.

Here, y_i shows the received message through the AWGN channel, while R_{ij} and L_{ij} show check node and variable node process, respectively. y_{fi} is the final code word calculated after maximum iteration.

The decoding process flow for LDPC-based coding is illustrated in Figure 6. Equations (2)–(6) provide the mathematical formulation for this process [21, 22]. To facilitate understanding of the iterative message passing decoding of LDPC codes, a Tanner graph approach is employed, as shown in Figure 7. By making simplifying independence assumptions, the messages in the message passing (MP) algorithm can be interpreted as the beliefs or estimated probabilities of the model regarding the values of each variable node in the graph [23]. The Tanner graph provides a graphical representation of the connections between variable nodes and check nodes in the LDPC code. It shows how the messages are passed iteratively between these nodes, facilitating comprehension of the iterative decoding process. The MP algorithm aims to refine and update these messages through multiple iterations, improving the accuracy of the decoded message.

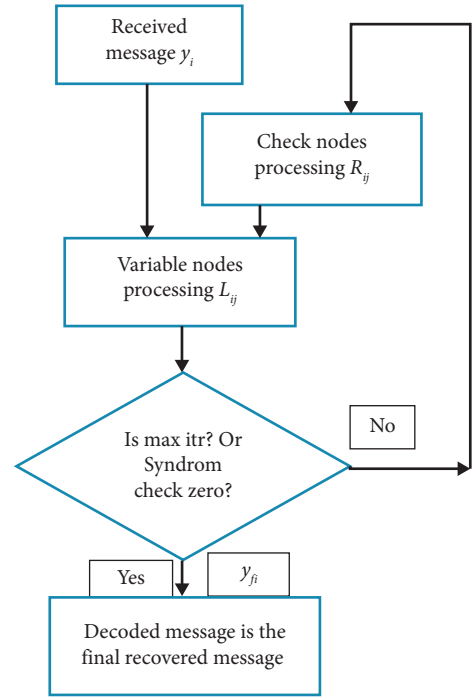

Figure 6: Decoding flow.

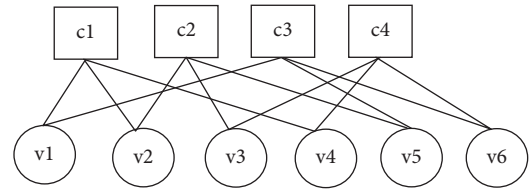

Figure 7: Tanner graph.

Initial LLR calculation is as follows:

$$\mathrm{LLR}_{y(i)} \cong \frac{2yi}{\sigma^2}. \tag{2}$$

Check node calculation is as follows:

$$R_{ij} = \left(\prod_{k\in v(i)/j} \mathrm{sign}(L_{ki})\right) \min_{k\in v(i)/j} L_{ki}. \tag{3}$$

Variable node calculation is as follows:

$$L_{ij} = \mathrm{LLR}_{yi} + \sum_{k\in c(i)/j} R_{ki}. \tag{4}$$

Final variable node calculation after completion of iteration is as follows:

$$yfi = \mathrm{sign}\left(\mathrm{LLR}_{(yi)} + \sum_{k\in c(i)/j} R_{ki}\right). \tag{5}$$

Hard decision is taken at the end of final calculation, and code bits have been recovered as follows:

$$c_i = \begin{cases} 0, & yfi = +1, \\ 1, & yfi = -1. \end{cases} \tag{6}$$

The computational complexity of the traditional decoder for an (N, K) regular LDPC code can be described using (7). In this equation, N represents the code length, K represents the number of information bits, w_r represents the row weight (number of nonzero elements per row) of the LDPC code, and w_c represents the column weight (number of nonzero elements per column) of the LDPC code. The complexity is measured in terms of the number of addition and comparison operations required for the decoding process. This information helps assess the computational requirements and efficiency of the traditional decoder for LDPC codes [5, 9].

$$2w_rN \text{ addition } \& \ 2w_cN \text{ comparision.} \tag{7}$$

2.4. Neural Network-Based Approach. There are various architectures of neural networks, which are classified based on the layer connections. One such architecture is the feedforward network (FFNN), also known as a multilayer perceptron [20]. In the FFNN architecture, the connections between layers are made only in a forward direction. Each layer in an FFNN consists of multiple interconnected neurons. In a neuron, the weighted inputs are summed together, an optional bias term is added, and the result is passed through a nonlinear activation function. The activation function can be either a sigmoid function or a rectified linear unit (ReLU) [18]. In the current work, the FFNN architecture has been utilized, and the ReLU activation function has been chosen. The ReLU activation function is a popular choice in many deep learning applications due to its ability to efficiently handle the vanishing gradient problem and its simplicity in computation. By using the FFNN architecture and the ReLU activation function, the current work leverages the capabilities of neural networks to decode LDPC codes effectively and achieve desired performance in the communication system.

The summary of the architecture used in the current work is shown in Table 1, while the flow of the NN design is illustrated in Figure 8.

In the current work of NN design, the neural network architecture comprises a total of 7 layers, which include the input layer, output layer, dropout layers, and dense layers. The selection of the hidden layers in the network is based on calculations outlined in the study in [24]. The depth of the network is determined using the formula $\log10(n)$, where n represents the sample size [21, 24]. Considering a sample size slightly above 10,000, the depth of the network falls between 4 and 5 layers. Therefore, in the current work, 5 inner hidden layers are chosen, which includes the dropout layer. The dropout layer is inserted between each hidden layer to mitigate the risk of overfitting in the network [21]. Overfitting occurs when the model becomes too specialized to the training data and performs poorly on unseen data. The dropout layer randomly selects a portion of neurons in each training iteration, temporarily excluding them from the network, thus preventing the network from relying too heavily on specific neurons or features. By including dropout layers and selecting an appropriate number of hidden layers based on the sample size, the current work aims to optimize the network's performance and generalization capabilities while reducing the risk of overfitting.

In the current work, the TensorFlow library with Keras as the frontend is utilized for the implementation of the neural network. The sequential model is employed in this approach, which allows for a linear stack of layers in the network. The received code is first passed to the input layer, where the weights are multiplied with the 54 neurons of the layer and biases are added to them. Then, the output is forwarded to the next layer through the rectified linear unit (ReLU) activation function, which introduces nonlinearity into the network and allows for better representation of complex patterns.

The operation performed by each neuron in a NN can be represented in the following equation:

$$y_i = \varphi_i\left(\sum_{j=1}^{i} w_j^i z_j^i + b^i\right), \tag{8}$$

where $\mathbf{y_i}$ is the excitation of node $\mathbf{i}$,$\mathbf{n^i}$ is the total incoming connections, $\mathbf{z^i}$ is the input, $\mathbf{w^i}$ is the weight, $\mathbf{b^i}$ is the bias, and $\boldsymbol{\varphi}_\mathbf{i}$ is the activation function at the $\mathbf{i^{th}}$ node to limit the amplitude of the output node. The input $\mathbf{z}$ is received at the neuron after multiplying with weight $\mathbf{w}$. The value of weight is always between 0 and 1 as the value fed at the input layer is the normalized value.

To prevent overfitting, a dropout layer is employed, where a certain percentage (e.g., 10%) of connections between neurons is randomly removed during each training iteration. This helps in improving the generalization capability of the network by preventing it from relying too heavily on specific connections. Finally, the output from the previous layers is passed to the output layer, which contains 54 neurons representing the encoded code. The hidden layers consist of 162 neurons in total. During the training phase, the weights and biases assigned to the connections between neurons are adjusted to minimize the difference between the predicted output and the desired output. By training the network with a sufficient amount of data, a more generalized model is created that can be used for predicting the encoded code for unseen inputs. In summary, the current work employs the TensorFlow library with Keras as the frontend, utilizing the sequential model. The input code is processed through multiple layers, including the input layer, hidden layers with ReLU activation, dropout layer, and output layer. The network is trained to optimize the weights and biases, resulting in a more generalized model that can predict the encoded code efficiently.

In the current work, the tuning of the network is performed by adjusting three key parameters: the learning rate, the number of epochs, and the datasets used for training. The learning rate determines the step size at which the weights and biases of the network are updated during the training process. By changing the learning rate, the speed and stability of the learning process can be controlled. The number of epochs represents the number of times the entire dataset is passed through the network during training. Increasing the number of epochs allows the network to learn more from the

Table 1: Summary of network architecture [21].

SN	Network architecture used		
	Name of the layers	Output dimensions	Activation function
1	Input layer (layer 1)	162	ReLu
2		Dropout layer (10%)	
3	Dense layer (layer 3)	162	ReLu
4		Dropout layer (10%)	
5	Dense layer (layer 5)	162	ReLu
6		Dropout layer (10%)	

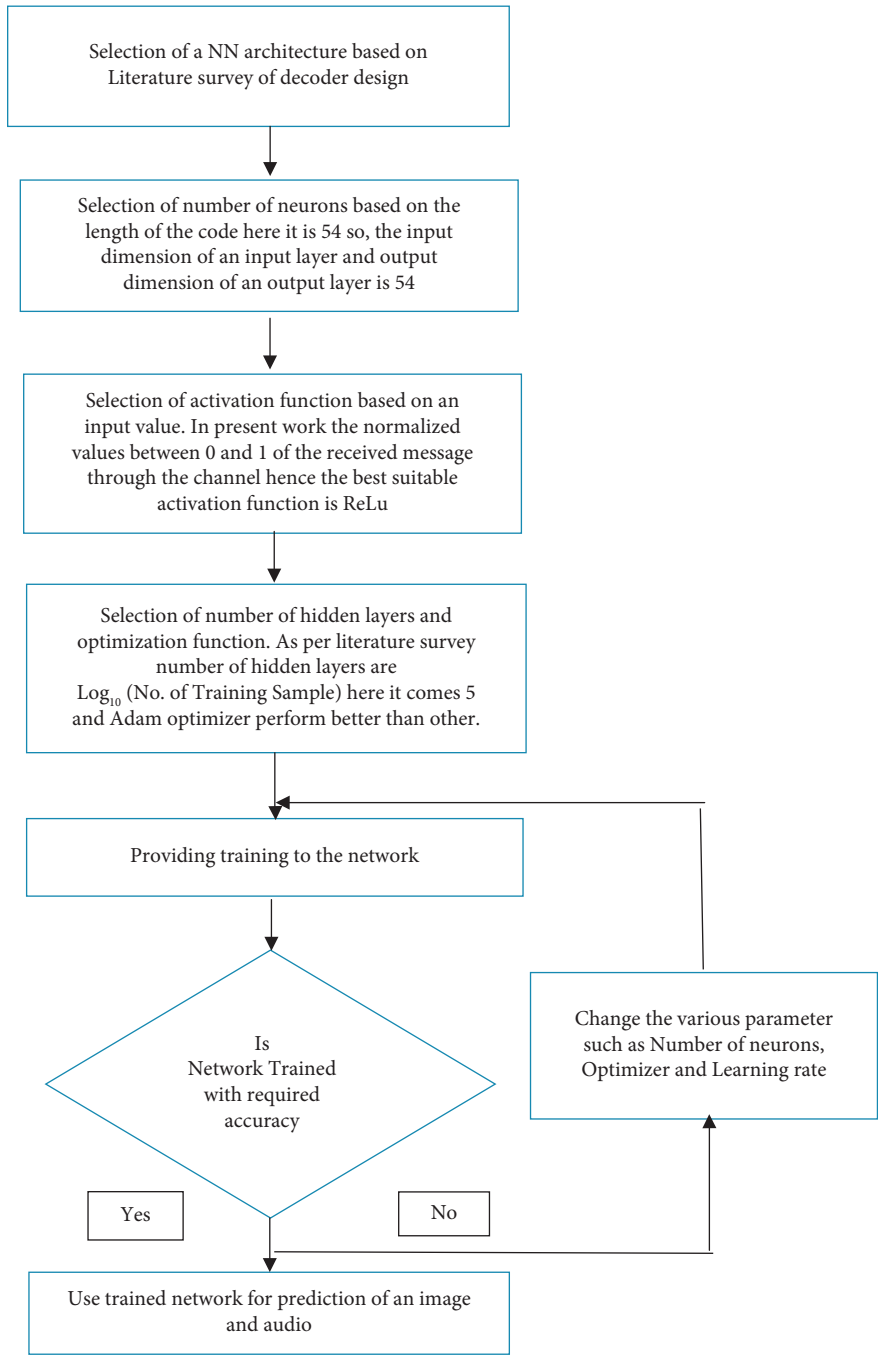

Figure 8: NN design flow.

data, potentially improving its performance. However, it is important to find a balance to avoid overfitting. Regarding the datasets used for training, in the current work, different types of training data are employed. Initially, the network is trained using an all-zero codeword, followed by a dataset consisting of half zeros and half ones, and finally, random combinations of zeros and ones are used. This approach allows the network to learn from different patterns and variations in the input data. According to the literature survey, one can use either an all-zero codeword or the complete code book for training the network. In the current work, a specific approach has been chosen for training. Based on the experimental results, the designed neural network in the current work achieves an accuracy of 70%, indicating the percentage of correct predictions made by the network. The loss, as observed from Figure 9, is 30%, representing the discrepancy between the predicted outputs and the desired outputs. It is important to minimize the loss to improve the accuracy and performance of the network. Overall, the tuning process in the current work involves adjusting the learning rate, the number of epochs, and utilizing different datasets for training, ultimately leading to a neural network with a 70% accuracy and a 30% loss.

The computational complexity of FFNN is given by equations (9), (10), and (11) [25].

$$\text{Real multiplication (RM)} = n_n n_i, \tag{9}$$

$$\text{Number of bit operation (BOP)} = n_n n_i \left[b_w b_i + A_{cc}\left(n_i, b_w, b_i\right)\right], \tag{10}$$

$$\text{Number of addition and bit shift (NABS) operation} = n_n n_i \left(X_w + 1\right) A_{cc}\left(n_i, b_w, b_i\right), \tag{11}$$

where n_i is the number of input feature, n_n is the number of neurons, b_i is the input bit width, b_w is the weight bit width, and X_w is the number of adders required to perform multiplication operation

Here, real multiplication indicates software complexity of the NN while the BOP and NABS operation indicate hardware complexity during implementation of NN on hardware.

3. Results

The simulation results of both the conventional decoder and the neural network (NN) decoder are presented in this analysis. These results are obtained through a comparative analysis of three audio samples of low-girth code. The audio samples used in this analysis are sourced from the NOIZEUS web sources [26], with the corpus description published in the study in [27]. According to the study in [27], the noisy signal samples used for training were taken from the AURORA database. These noisy signals were combined with speech signals at different signal-to-noise ratio (SNR) levels. In this analysis, three audio samples with a 5 dB SNR were selected for evaluation. These noisy signal samples of audio signals are visually represented by converting it from wave format to periodograms [26]. By comparing the performance of the conventional decoder and the neural network decoder on these low-girth code audio samples, the effectiveness of the NN decoder can be assessed. The simulation results provide insights into the performance, accuracy, and reliability of both decoding approaches, allowing for a comparative analysis of their capabilities. Overall, the simulation results obtained through the evaluation of these audio samples provide valuable information on the performance of the conventional and NN decoders for low-girth codes, contributing to the understanding of their comparative performance and potential applications.

3.1. Performance Comparison with Conventional Method. The comparative analysis in terms of BER, PSNR, and MSE is shown here between conventional and NN decoders. Figures 10–12 are the periodogram representation of audio sample taken from NOIZEUS web sources [26] (Tables 2–4).

The BER plots shown in Figures 13–15 represent three samples. These plots demonstrate the presence of an error floor in the conventional decoder for low-girth codes, whereas no error floor is observed in the NN decoder. It is observed that BER increases with higher SNR values. For all three samples, BER obtained using conventional decoders is approximately 1×10^{-1} at an SNR of 7 dB, while for the NN decoder, it is 8×10^{-4} at the same SNR value of 7 dB.

The PSNR plots shown in Figures 16–18 represent three samples. These plots indicate that there is no increase in the PSNR value for the conventional decoder, even with a rise in SNR value. However, for the NN decoder, the PSNR value increases with higher SNR values. For all three samples, PSNR obtained using conventional decoders is approximately 39 dB at an SNR of 7 dB, while for the NN decoder, it is 41 dB at the same SNR value of 7 dB.

The mean square error (MSE) plots shown in Figures 19–21 represent three samples. It is observed that as SNR increases, the error reduces and approaches zero for the NN decoder. However, for the conventional decoder, the error remains high for low-girth codes. MSE obtained using the conventional decoder is 94% at an SNR of 7 dB, while for the NN decoder, it is approximately 6% at the same SNR value of 7 dB for all three samples taken.

4. Discussion

The bit error rate (BER) represents the number of bit errors per unit time. It is calculated as the ratio of the number of erroneous bits to the total number of transmitted bits during a specific time interval. The BER is a unitless measure of

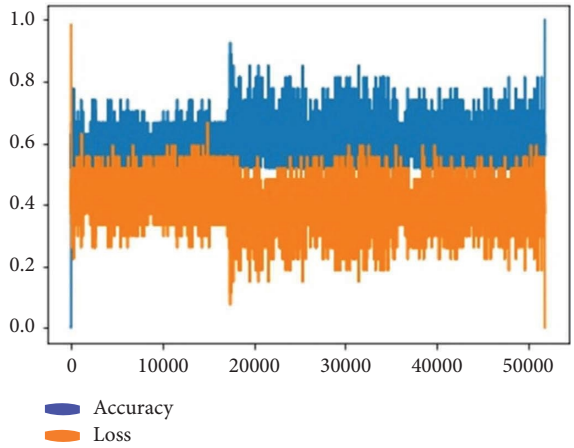

Figure 9: Accuracy and loss of trained NN.

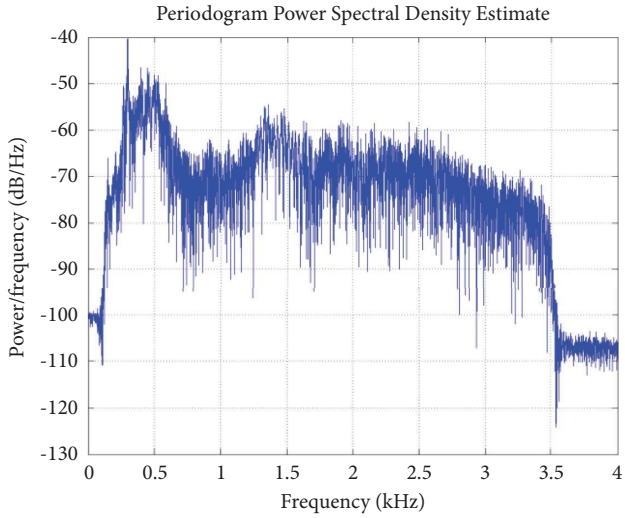

Figure 10: Original audio sample 1 of train noise [26].

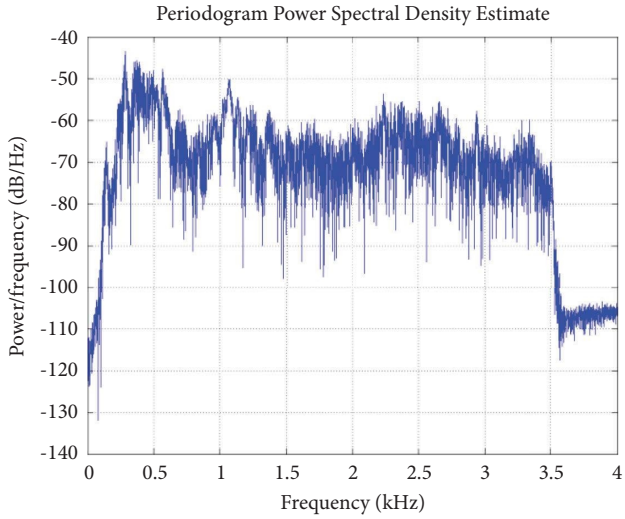

Figure 11: Original audio sample 2 of train noise [26].

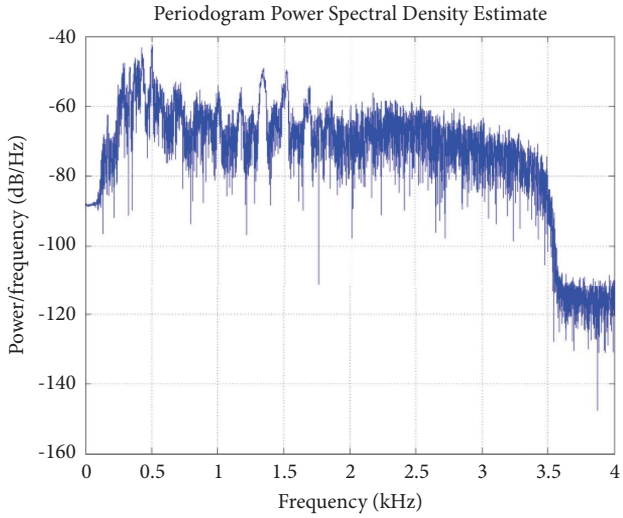

Figure 12: Original audio of sample 3 of train noise [26].

performance often expressed as a percentage [28]. In the BER plot, there is a point at which the curve does not decrease as rapidly as before, indicating a region where performance levels off. This region is referred to as the error floor region. The portion of the plot just before the sudden drop in performance is known as the waterfall region [29]. A decoder's performance is considered high when there is no error floor region and a well-defined waterfall region in the BER plot.

The low-girth code introduces an error floor in conventional decoding, which is evident in Figures 13–15 of the BER plot for the conventional decoder. In contrast, the results obtained for the NN decoder do not exhibit an error floor in the decoding process. This can be observed from the same Figures 13–15.

The PSNR (peak signal-to-noise ratio) is a widely used metric to assess the quality of reconstructed data in lossy compression codecs. In the case of 8 bit data, typical PSNR values for lossy image and video compression range between 30 and 50 dB. For 16 bit data, typical PSNR values fall between 60 and 80 dB [30, 31]. Acceptable PSNR values for wireless transmission quality loss are generally around 20 to 25 dB [32, 33]. In the current study, the achieved PSNR is approximately 41 dB at an SNR of 7 dB and 96 dB at an SNR of 10 dB which is observed from the PSNR plot of Figures 16–18.

NN developed here is the best suitable decoder for low-girth codes compared to conventional decoders.

BER of 8×10^{-4}, obtained in the study in [3] and shown in Figure 22, is achieved at an SNR of 10 dB. However, the current decoder demonstrates a coding gain of 3 dB in terms of BER performance compared to the study in [3]. In the current work, the same BER of 8×10^{-4} is achieved at an SNR of 7 dB. Furthermore, Figure 23 provides a comparison in terms of the number of epochs used during the training of the network. In the study in [18], a BER of 10^{-4} is obtained at an SNR of 7 dB by training the network for 16 epochs. In

Table 2: Obtained BER, PSNR, and MSE of low-girth audio sample 1 of train noise of conventional and neural network decoders [21].

SNR (dB)	BER conventional decoder	BER NN decoder	PSNR conventional decoder	PSNR NN decoder	MSE conventional decoder	MSE NN decoder
1	0.872898	0.05667	39.71535	40.25115	0.949897	0.467508
7	0.898795	0.000813	39.71734	41.58944	0.947471	0.064932
8	0.903513	0.000191	39.7174	42.03299	0.947397	0.03147
9	0.903389	$4.21E-05$	39.71855	42.60724	0.945998	0.011642
10	0.907868	0	39.71591	96.32947	0.949216	0
11	0.913214	0	39.71892	96.32947	0.945555	0

Table 3: Obtained BER, PSNR, and MSE of low-girth audio sample 2 of train noise of conventional and neural network decoders [21].

SNR (dB)	BER conventional decoder	BER NN decoder	PSNR conventional decoder	PSNR NN decoder	MSE conventional decoder	MSE NN decoder
1	0.872117	0.062062	39.71493	40.24071	0.950409	0.474213
7	0.899559	0.000852	39.71521	41.55545	0.950066	0.068534
8	0.901092	0.000454	39.7154	41.977	0.949834	0.034554
9	0.906769	$3.56E-05$	39.71481	42.72688	0.950552	0.009385
10	0.910385	0	39.71606	96.32947	0.949026	0
11	0.913172	0	39.71561	96.32947	0.949583	0

Table 4: Obtained BER, PSNR, and MSE of low-girth audio sample 3 of train noise of conventional and neural network decoders [21].

SNR (dB)	BER conventional decoder	BER NN decoder	PSNR conventional decoder	PSNR NN decoder	MSE conventional decoder	MSE NN decoder
1	0.872454	0.056285	39.70957	40.25216	0.956956	0.466867
7	0.900528	0.000732	39.71099	41.74078	0.955217	0.050925
8	0.904614	0.000214	39.71115	41.96716	0.955029	0.035124
9	0.904491	$2.92E-05$	39.71028	44.3992	0.956092	0.000331
10	0.909436	0	39.70917	96.32947	0.957452	0
11	0.911195	0	39.71127	96.32947	0.954872	0

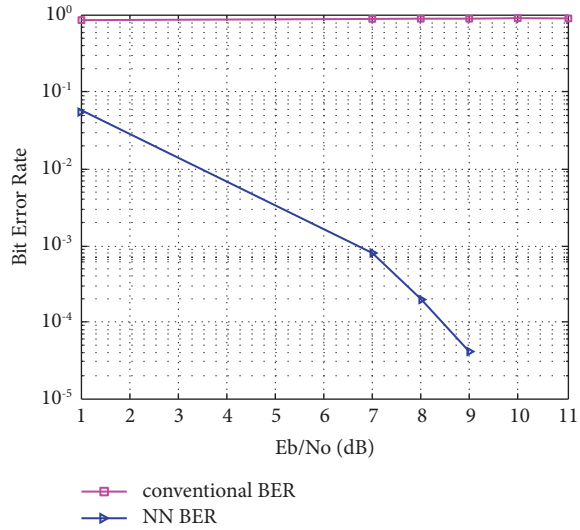

Figure 13: BER comparison of sample 1 of train noise.

contrast, the current work achieves the same BER at the same SNR using only 5 epochs of training.

The BER performance of the conventional decoder exhibits an error floor for low-girth codes. This error floor occurs because the same type of information is exchanged between variable nodes and check nodes during the iterative decoding process. As a result, BER of the conventional decoder for low-girth codes remains at 10^{-1} for all SNRs.

The simulation results of the NN decoder demonstrate that the BER performance improves as the SNR increases for

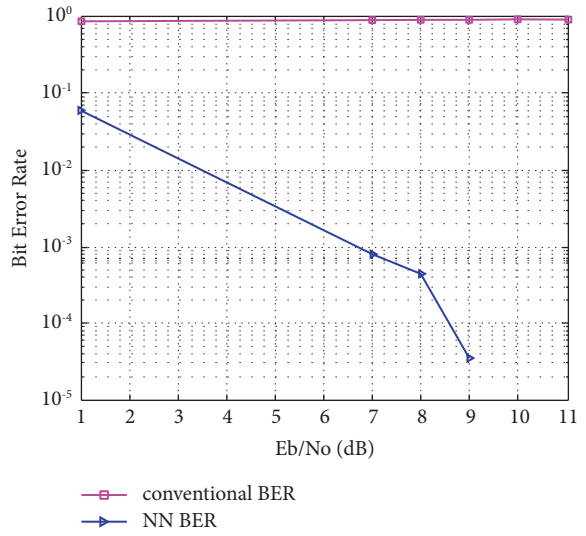

Figure 14: BER comparison of sample 2 of train noise.

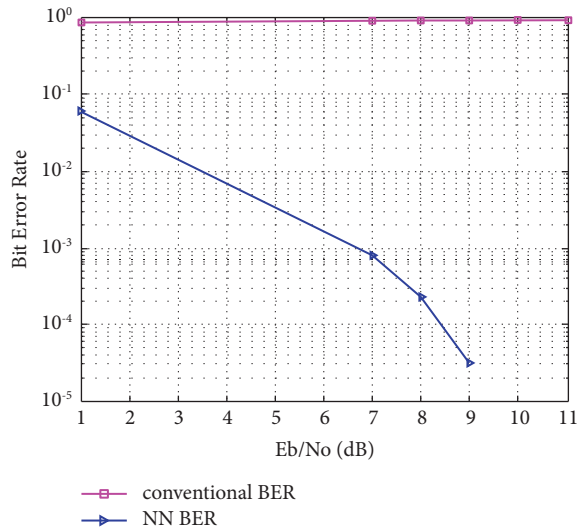

Figure 15: BER comparison of sample 3 train noise.

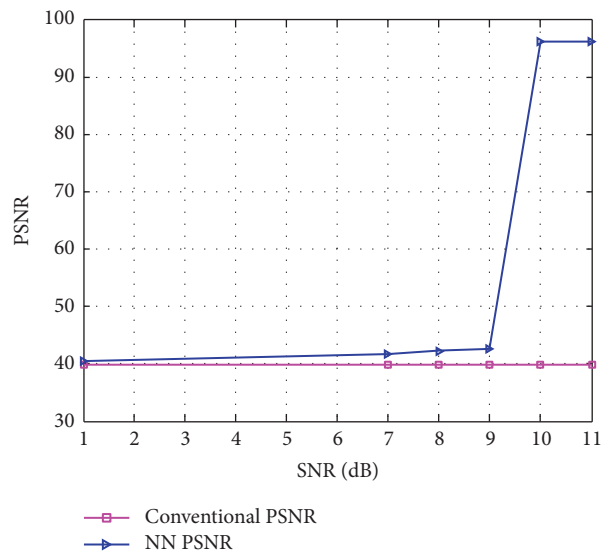

Figure 16: PSNR comparison of sample 1 of train noise.

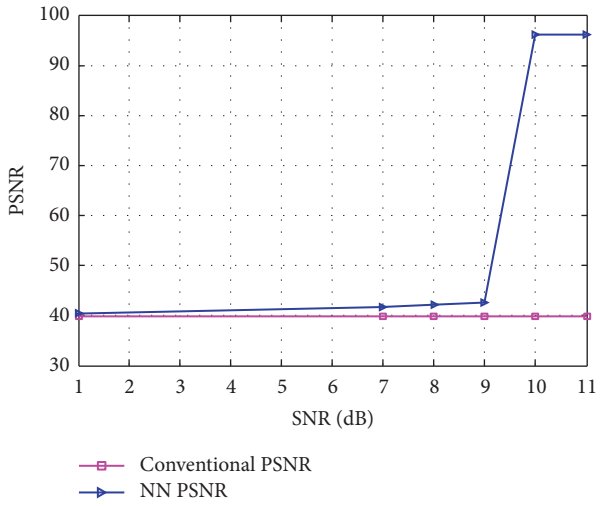

Figure 17: PSNR comparison of sample 2 of train noise.

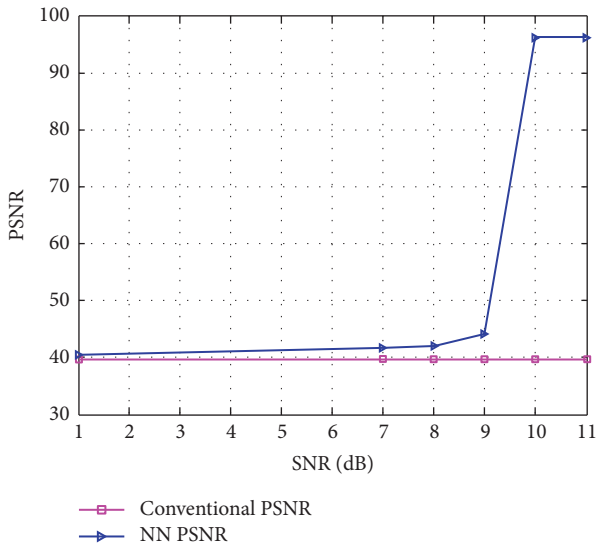

Figure 18: PSNR comparison of sample 3 of train noise.

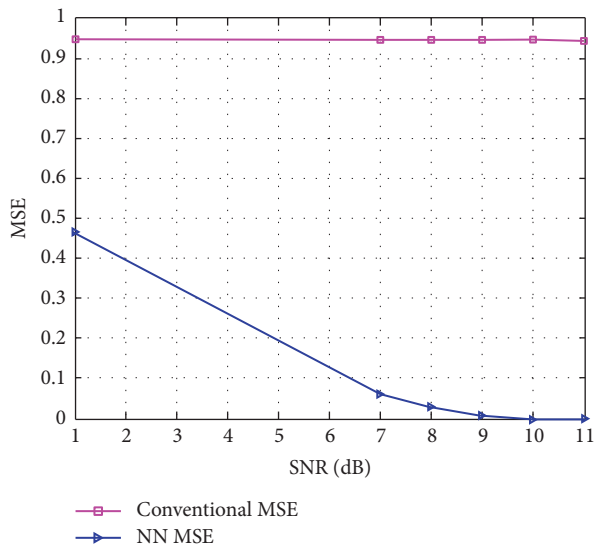

Figure 19: MSE comparison of sample 1 of train noise.

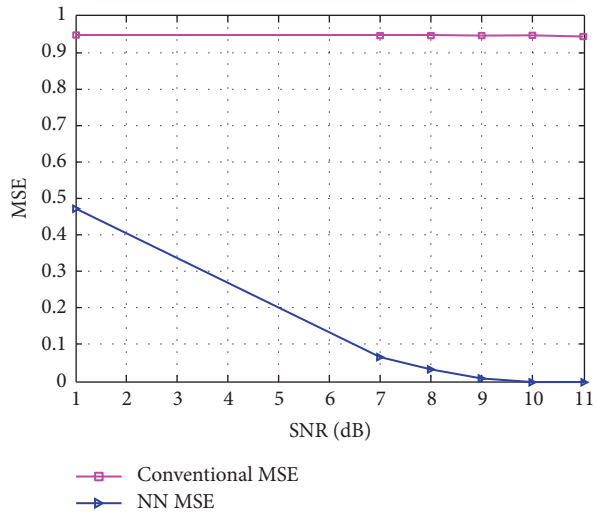

Figure 20: MSE comparison of sample 2 of train noise.

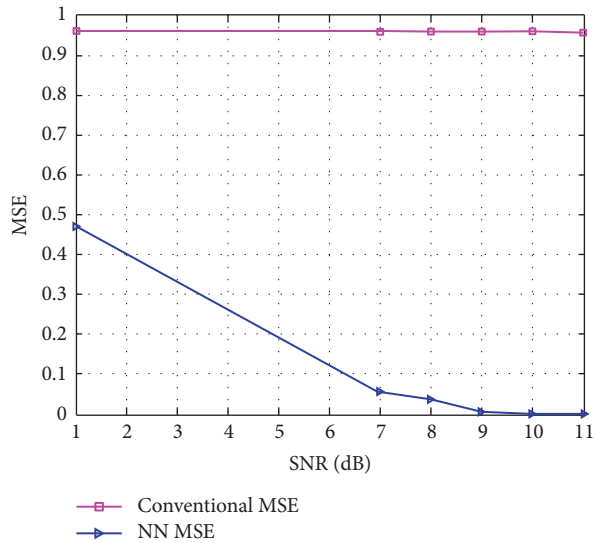

Figure 21: MSE comparison of sample 3 train noise.

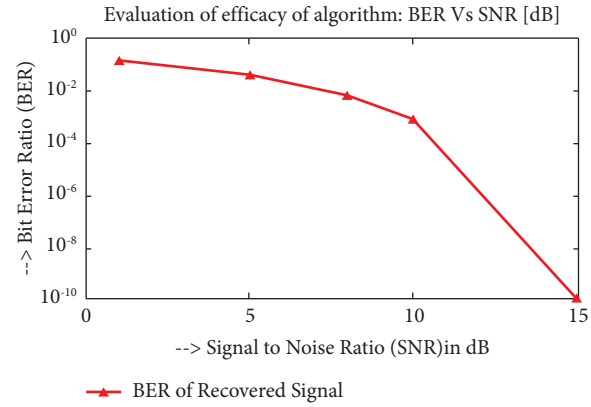

Figure 22: Obtained BER conventional decoder [3].

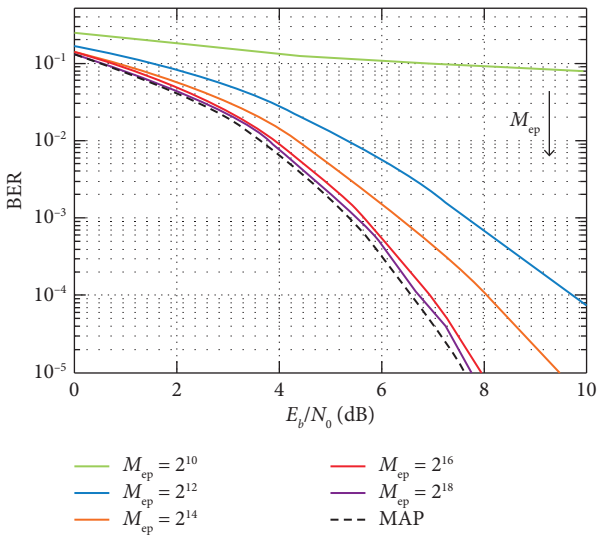

Figure 23: Obtained BER deep learning-based decoding [18].

the low-girth code. In addition, it is observed that PSNR of the decoded images also increases with the increasing BER of the decoded code.

In the absence of noise, the two audios are identical, and thus MSE is zero. In this case, PSNR is maximum [34].

5. Conclusions

The results obtained using the conventional algorithm developed in the current work show a BER of 10^{-1} for a signal-to-noise ratio of 11 dB for the low-girth code. In contrast, a significantly lower BER of 10^{-4} is achieved using the NN-based decoder at a signal-to-noise ratio of 7 dB. These results are depicted in Figures 13–15. This significant improvement in BER using the NN-based decoder leads to a power saving of approximately 30 to 40%. By utilizing such a decoder, the life span of the power system can be extended and communication errors can be significantly reduced.

The decoder developed here effectively removes the noise introduced by the channel, as verified by the mean square error plot shown in Figures 19–21. The advantage of the NN-based decoder is that it is not constrained by the limitations of high-girth code construction. This is evident from the results obtained for low-girth codes, where no error floor is observed. The performance of the developed decoder is superior for low-girth codes, while the conventional decoder fails to converge for such codes, as confirmed by the bit error rate plot. Furthermore, the performance of the NN-based decoder improves with increasing SNR.

In the current work, the conventional decoder was specifically developed for a low-girth code. As a result, the achieved BER value of 10^{-1} reflects the limitations of the low-girth code. However, it is possible to improve the BER performance by developing a similar decoder using a high-girth code. Higher girth codes can offer better error correction capabilities and potentially yield lower BER values.

In comparison to the best conventional decoder developed by the authors in [3], the NN-based decoder developed here demonstrates a coding gain of 3 dB. The conventional decoder is unable to perform well for low-girth codes, as evidenced by the generated error floor observed in the bit error rate plot. On the other hand, the NN-based decoder achieves a lower BER of 10^{-4} at an assumed SNR of 10 dB. The quality of the recovered audio is considered good, as confirmed by the peak signal-to-noise ratio plot. Furthermore, it was observed that the trained network is capable of decoding the code using a single iteration, whereas the conventional decoder requires multiple iterations. This highlights the efficiency of the NN-based decoder in terms of computational speed and performance.

The present NN decoder is better than the conventional decoder in the following ways:

(i) The NN decoder offers a significant advantage over the conventional method by enabling the decoding of low-girth LDPC codes. Unlike the conventional decoder, which generates an error floor for such codes, the NN decoder is not constrained by the code's girth and delivers consistent performance for both low- and high-girth codes. This capability of the NN decoder to handle different code structures effectively eliminates the limitations imposed by the girth of the code on the decoding process.

(ii) In contrast to the conventional decoder, which typically requires multiple iterations in the decoding process, the NN decoder is capable of decoding the code using a single iteration. This is a significant advantage of the NN decoder as it reduces the computational complexity and decoding latency. By leveraging the power of neural networks, the NN decoder can effectively capture the decoding patterns and make accurate decisions in a single pass, thereby enhancing the efficiency of the decoding process.

(iii) NN decoder performance improves further by tuning the learning parameter such as optimizer, neurons, and no. of hidden layers.

Data Availability

It is to be state that the noisy audio sample are taken from the website https://ecs.utdallas.edu/loizou/speech/noizeus/ which is freely available on the internet. These audio samples are used for encoding and decoding using conventional and NN-based approach. The authors have taken audio samples as raw data from the abovementioned website. These approaches are developed in this current work.

Conflicts of Interest

The authors Mr. Dharmeshkumar Jayantibhai Patel and Dr. Ninad Sunilkumar Bhatt certify that they have no affiliation with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing

arrangements) or nonfinancial interest (such as personal or professional relationships, affiliations, knowledge, or beliefs) in the subject matter or materials discussed in this manuscript.

Acknowledgments

The authors would like to thank Dr. N. S. Bhatt, Professor, and their research supervisors for their patient instruction, passionate support, and constructive criticisms of this research work. The authors would like to thank Prof. R. N. Mehta, Associate Professor, and Prof. P. J. Patel, Assistant Professor, for providing language help. The authors would also like to thank Dr. Pinalkumar, Engineer, for providing guidance during technical issues.

References

[1] J. C. David and MacKay, *Information Theory, Inference and Learning Algorithms*, CUP, Cambridge, UK, 2003.

[2] C. Gao, S. Liu, D. Jiang, and L. Chen, "Constructing ldpc codes with any desired girth," *Sensors*, vol. 21, no. 6, pp. 2012–2014, 2021.

[3] J. Baviskar, A. Mulla, M. Gulati, P. Vaswani, and A. Baviskar, "LDPC based error resilient audio signal processing for wireless communication," in *Proceedings of the 2015 International Conference on Pervasive Computing (ICPC)*, Pune, India, January 2015.

[4] Neha, "Implementation and performance analysis of convolution error correcting codes with code rate=1/2," in *Proceedings of the 2016International Conference on Micro-Electronics and Telecommunication Engineering*, pp. 482–486, ICMETE, Ghaziabad, India, September 2016.

[5] J. Li, N. Di, P. Gao, G. Sheng, and Z. Wu, "A digital audio broadcasting system using short length QC-LDPC," in *Proceedings of the 2012 IEEE 14th International Conference on Communication Technology*, Chengdu, China, November 2012.

[6] Y. Wang, Z. Zhang, S. Zhang, S. Cao, and S. Xu, "A unified deep learning based polar-LDPC decoder for 5G communication systems," in *Proceedings of the 2018 10th International Conference on Wireless Communications and Signal Processing*, pp. 1–6, WCSP, Hangzhou, China, October 2018.

[7] J. Zhao, F. Zarkeshvari, and A. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Transactions on Communications*, vol. 53, no. 4, pp. 549–554, 2005.

[8] J. Fan and Y. Xiao, "A method of counting the number of cycles in LDPC codes," in *Proceedings of the 2006 8th international Conference on Signal Processing*, pp. 4–7, Guilin, China, November 2006.

[9] A. Kalsi, A. Bajpai, L. Wuttisittikulkij, and P. Kovintaewat, "A base matrix method to construct column weight 3 quasi-cyclic LDPC codes with high girth," in *Proceedings of the 2016 International Conference on Electronics, Information, and Communications*, pp. 1–4, (ICEIC), Danang, Vietnam, January 2016.

[10] Y. He, J. Yang, and J. Song, "A survey of error floor of LDPC codes," in *Proceedings of the 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, pp. 61–64, Harbin, China, August 2011.

[11] C.-F. Teng, H.-M. Ou, and A.-Y. A. Wu, "Neural network-based equalizer by utilizing coding gain in advance," in *Proceedings of the 2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1–5, IEEE, Ottawa, Canada, November 2019.

[12] M. Helmling, E. Rosnes, S. Ruzika, and S. Scholl, "Efficient maximum-likelihood decoding of linear block codes on binary memoryless channels," in *Proceedings of the 2014 IEEE International Symposium on Information Theory*, pp. 2589–2593, IEEE, Honolulu, HI, USA, July 2014.

[13] I. Ortuno, M. Ortuno, and J. A. Delgado, "Error correcting neural networks for channels with Gaussian noise," in *Proceedings of the [Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 4, pp. 295–300, Baltimore, MD, USA, June 1992.

[14] M. Rafiq, G. Bugmann, and D. Easterbrook, "Neural network design for engineering applications," *Computers & Structures*, vol. 79, no. 17, pp. 1541–1552, 2001.

[15] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1361–1365, IEEE, Aachen, Germany, June 2017.

[16] A. Karami, M. A. Attari, and H. Tavakoli, "Multi-layer perceptron neural networks decoder for ldpc codes," in *Proceedings of the 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, IEEE, Beijing, China, September 2009.

[17] E. Nachmani, Y. Be'Ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proceedings of 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 341–346, Monticello, IL, USA, September 2026.

[18] T. Gruber, S. Cammerer, J. Hoydis, and S. Ten Brink, "On deep learning-based channel decoding," in *Proceedings of the 2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, arXiv, Baltimore, MD, USA, March 2017.

[19] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Beery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 1–13, 2017.

[20] J. Seo, J. Lee, and K. Kim, "Decoding of polar code by using deep feed-forward neural networks," in *Proceedings of the 2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 238–242, Maui, HI, USA, March 2018.

[21] D. Patel and N. Bhatt, "A neural network-based approach for decoding of an image of low-girth LDPC code," *International Journal of Engineering Trends and Technology*, vol. 70, no. 11, pp. 305–314, 2022.

[22] D. J. Patel, P. Engineer, and N. S. Bhatt, "Image communication using quasi-cyclic low-density parity-check(qc-ldpc) code," *Advances in VLSI and Embedded Systems*, Springer, Singapore, pp. 211–221, 2021.

[23] A. Payani and F. Fekri, "Decoding LDPC codes on binary erasure channels using deep recurrent neural-logic layers," in *Proceedings of the 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC)*, pp. 1–5, Hong Kong, China, December 2018.

[24] M. H. Farrell, T. Liang, and S. Misra, "Deep neural networks for estimation and inference," *Econometrica*, vol. 89, no. 1, pp. 181–213, 2021.

[25] P. J. Freire, S. Srivallapanondh, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn, "Computational complexity evaluation of

neural network applications in signal processing," 2022, http://arxiv.org/abs/2206.1219.

[26] ECS, "NOIZEUS: A noisy speech corpus for evaluation of speech enhancement algorithms," 2007, https://ecs.utdallas.edu/loizou/speech/noizeus/.

[27] H. Hirsch and D. Pearce, *The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions*, ISCA ITRW ASR, Paris, France, 2000.

[28] J. Lim, *Is BER the Bit Error Ratio or the Bit Error Rate?*, EDN, Colorado, CO, USA, 2010.

[29] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*, Cambridge University Press, Cambridge, UK, 2009.

[30] S. T. Welstead, *Fractal and Wavelet Image Compression Techniques*, SPIE Publication, Washington, DC, USA, 1999.

[31] R. Hamzaoui and D. Saupe, "Fractal image compression," in *Document and Image Compression*, M. Barni, Ed., CRC Press, Boca Raton, FL, USA, 2006.

[32] N. Thomos, N. V. Boulgouris, and M. G. Strintzis, "Optimized transmission of JPEG2000 streams over wireless channels," *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 54–67, 2006.

[33] L. Xiangjun and C. Jianfei, "Robust transmission of jpeg2000 encoded images over packet loss channels," in *Proceedings of the 2007 IEEE International Conference on Multimedia and Expo*, pp. 947–950, Beijing, China, July 2007.

[34] Salomon and David, *Data Compression: The Complete Reference*, Springer, Singapore, 4 edition, 2007.